Synthetic Aerial Image Generation for Miniature Aerial System

Srikanth A[1], L Krishnamurthy[1]
[1]Dept. of Mechanical Engineering
The National Institute of Engineering
Mysuru, India

L P Prathyusha[2]
[2]Dept. of Avionics
Jawaharlal Nehru Tech. University
Kakinada, India

VPS Naidu[3]
[3]CSIR – NAL
Bengaluru, India
vpsnaidu@gmail.com

***Abstract*— This paper presents a computer vision based method which generates a synthetic image of the earth as would be viewed by an aerial camera. The method takes geo-referenced, ortho-rectified aerial image database as the source and using a pinhole perspective camera model generates the synthetic image. The method requires the position and attitude of the camera, which act as the extrinsic parameters for the camera model. The intrinsic parameters are chosen to emulate a real camera. Finally, the results of synthetic aerial image generation implemented on MATLAB are presented.**

Keywords*— *Image Processing, Computer Vision, Synthetic Aerial Image Generation, Perspective Camera Model, Miniature Aerial System

I. INTRODUCTION

In today's world miniature aerial systems (MASs) have been experiencing tremendous growth and are attracting enormous interest and attention from the government, researchers, military, corporations and the hobbyists alike. Miniature aerial vehicles (MAVs) are a great tool for scientific research owing to their low cost, high manoeuvrability, and ease of maintenance [1]. Civilian applications of MAVs include target tracking, emergency monitoring, victim search and rescue, traffic monitoring, aerial filming, geological survey, weather forecast, radiation monitoring, pollution detection, etc. For defence applications, MAVs are an ideal choice for collecting information from close-range during intelligence, surveillance and reconnaissance missions. MAVs are highly suitable and are generally efficient at performing tasks that involve risk and repetition, i.e., what the military calls "dull, dirt and danger" which humans don't want to do. The goal in many of these tasks is to capture the image of an object for tracking and information-gathering purposes. One of the requirements for these applications is accurate detection and localization of the target. Target detection is the process of establishing the presence of the target of interest. Target geo-localization is the process of determining the geo-location (geographic location) of a target by knowing the pixel location of the target in the image captured by a vision sensor. In many cases, it may not be feasible to acquire images from a vision sensor mounted on a real vehicle due to cost and time constraints. This paper presents a method for generating aerial image/scene from a local database of GeoTIFF images when a vision sensor equipped MAV is not available for access. The focus is on generating the scene for any given pose configuration of the vision sensor. In real-time, the synthetic aerial images so generated can even serve as a source of reference to validate and further analyse the results of geo-localization.

II. METHODOLOGY

The method adopted for synthetic aerial image generation in this paper is derived from the technique for target geo-localization presented in [2]. This section describes the mathematics required for synthetic scene generation in detail.

A. Co-ordinate Frames

Firstly, we fix the co-ordinate frames associated with synthetic scene generation namely: inertial frame, gimbal frame, camera frame and image frame. The co-ordinate frames assignments are depicted in Figure 1 and the associated subscripts are given in Table I.

TABLE I. CO-ORDINATE FRAME SUBSCRIPTS ASSOCIATED- WITH SYNTHETIC AERIAL IMAGE GENERATION

Co-ordinate Frame	Subscript
Image frame (pixels)	*ip*
Image frame (metres)	*im*
Camera frame	*c*
Gimbal frame	*g*
World frame / Inertial frame	*i*

- Inertial frame (X_i, Y_i, Z_i), is fixed on the earth's surface with X_i directed East, Y_i directed North and Z_i directed toward the sky.
- Gimbal frame (X_g, Y_g, Z_g), originates at the gimbals' centre of rotation and is oriented so that Z_g points along the optical axis, Y_g points up the image plane, X_g points left in the image plane.
- Camera frame (X_c, Y_c, Z_c), originates at the optical center with X_c pointing left in the image plane, Y_c pointing up in the image plane and Z_c directed along the optical axis.
- The image frame is two-dimensional and is also called the image plane. The image frame co-ordinates are measured either in pixels (x_{ip}, y_{ip}) which originate at the top-left corner of the image or in meters $(x_{im},$

978-1-4673-6667-0/15/$31.00©2015 IEEE

y_{im}) which originate at the principal point of the camera.

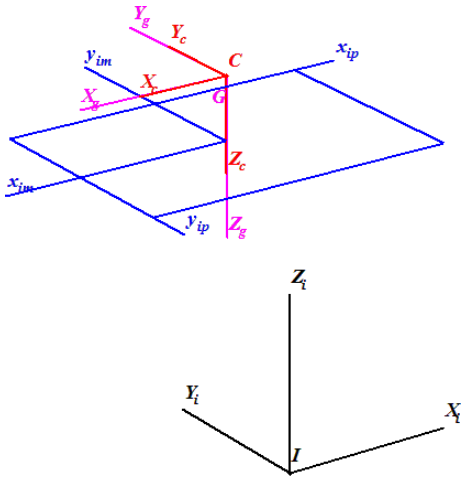


Fig. 1. Co-ordinate frames associated with synthetic aerial image generation

B. Transformations

There are two associated transformations: one between the inertial frame and the gimbal frame; the other between the gimbal frame and the camera frame as described in Table II.

TABLE II. HOMOGENEOUS TRANSFORMATION MATRICES ASSOCIATED WITH SYNTHETIC SCENE GENERATION

Homogeneous transformation matrix	Description
${}^{g}T_{i}$	Inertial Frame to Gimbal Frame Transformation
${}^{c}T_{g}$	Gimbal Frame to Camera Frame Transformation

1) Inertial Frame to Gimbal Frame Transformation

This transformation involves location of the inertial frame with respect to the gimbals' rotation centre and the rotation that aligns the gimbals' co-ordinate frame with the inertial frame.

$$ {}^{g}d_{i} = \begin{bmatrix} -{}^{i}X_{g} \\ -{}^{i}Y_{g} \\ -{}^{i}Z_{g} \end{bmatrix} \tag{1} $$

$$ {}^{g}R_{i} = R_{x}(\alpha)\cdot R_{y}(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ \sin\alpha\cdot\sin\beta & \cos\alpha & -\sin\alpha\cdot\cos\beta \\ \cos\alpha\cdot\sin\beta & -\sin\alpha & \cos\alpha\cdot\cos\beta \end{bmatrix} \tag{2} $$

$$ {}^{g}T_{i} = \begin{bmatrix} {}^{g}R_{i} & {}^{g}d_{i} \\ 0 & 1 \end{bmatrix} \tag{3} $$

${}^{g}d_{i}$ is a vector denoting the location of inertial frame with respect to the gimbals' rotation centre.

α is the tilt angle about ${}^{i}X_{g}$ and β is the pan angle about ${}^{i}Y_{g}$ after α.

2) Gimbal Frame to Camera Frame Transformation

This transformation depends on the location of the gimbals' rotation centre relative to the location of the camera's optical centre and the fixed rotation that aligns gimbal frame with that of the camera. In the MATLAB implementation that we have presented, the translations from gimbal frame to camera frame is assumed to be $[0\ 0\ 0]^{T}$.

$$ {}^{c}d_{g} = \begin{bmatrix} -{}^{g}X_{c} \\ -{}^{g}Y_{c} \\ -{}^{g}Z_{c} \end{bmatrix} \tag{4} $$

$$ {}^{c}R_{g} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5} $$

$$ {}^{c}T_{g} = \begin{bmatrix} {}^{c}R_{g} & {}^{c}d_{g} \\ 0 & 1 \end{bmatrix} \tag{6} $$

3) Camera Model

The process of image formation is analogous to tracing of rays from points on objects to pixels in the image [3]. A perspective camera projects point on an object $p_{obj} = [p_x\ p_y\ p_z\ 1]^{T}$ onto a point $q = [x_{ip}\ y_{ip}\ 1\ 1]^{T}$ in the image plane, where ${}^{c}p_{obj}$ denotes the inertial co-ordinate of a point relative to the centre of the camera (Figure 2).

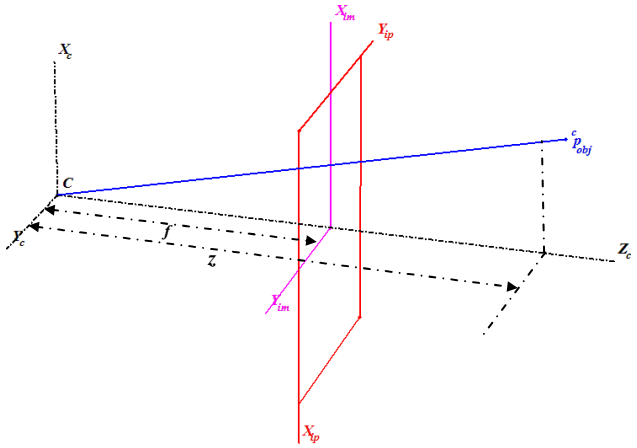


Fig. 2. A perspective camera model

The projection of the point on the object ${}^{c}p_{obj}$ in the image frame can be expressed in meters as

$$ \left.\begin{aligned} x_{im} &= (c_y + y_{ip})s_x \\ y_{im} &= (c_x + y_{ip})s_y \end{aligned}\right\} \tag{7} $$

where,

the units of (x_{ip}, y_{ip}) are pixels

the units of (x_{im}, y_{im}) are meters

(c_x, c_y) are the principal point of the camera in pixels

(s_x, s_y) are the image pixel scale factors

By similar triangles we get that $x_{im} = f\frac{p_x}{p_z}, y_{im} = f\frac{p_y}{p_z}$ where f is the focal length of the camera.

Defining $\lambda \approx p_z$ we get

$$\Lambda \cdot q = \begin{bmatrix} f_x & f_\theta & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot {}^{c}p_{obj} \tag{8}$$

$$\Lambda \cdot q = C \cdot {}^{c}p_{obj} \tag{9}$$

where,

$f_x = f/s_x$ is size of unit length in horizontal pixels,

$f_y = f/s_y$ is size of unit length in vertical pixels,

$f_\theta = f/s_\theta$ is skew of the pixel (often close to zero),

$\Lambda = \begin{bmatrix} \lambda I & 0 \\ 0 & 1 \end{bmatrix}$ and

C is known as the camera calibration matrix.

4) Pixel co-ordinates to World co-ordinates

The world co-ordinate ${}^{i}p_{obj}$ corresponding to a pixel co-ordinate point q is computed as follows:

Substituting for ${}^{c}p_{obj}$ in (9) we have

$$(\Lambda \cdot q) = C \cdot {}^{c}T_g \cdot {}^{g}T_i \cdot {}^{i}p_{obj} \tag{10}$$

Solving for the inertial co-ordinate of the point (${}^{i}p_{obj}$) gives

$${}^{i}p_{obj} = \left[C \cdot {}^{c}T_g \cdot {}^{g}T_i\right]^{-1} \cdot (\Lambda \cdot q) \tag{11}$$

Then, ${}^{i}p_{obj}$ can be determined when image depth, λ is known. The image depth λ can be calculated as follows [2]:

$$\lambda = \frac{{}^{i}z_{cc}}{{}^{i}z_{cc} - {}^{i}\bar{z}_{obj}} \tag{12}$$

III. Algorithm

Using the equations already described in section II the synthetic aerial image is generated. The details of the algorithm are discussed in this section.

1) Inputs to the algorithm

- Intrinsic parameters: focal length, pixel scale factor, principal point and the image dimensions. They are directly related to the emulated vision sensor and entered by the user using the interface.
- Extrinsic parameters: latitude and longitude of gimbal centre of rotation, altitude from ground level, pan/azimuth and tilt/elevation. The GPS co-ordinates in latitude and longitude are converted to UTM (Universal Transverse Mercator) co-ordinates which are in metres.
- GeoTIFF images for the geographical area considered: The GeoTIFF images are obtained using steps described in Appendix A.

2) Output from the algorithm

The output from the algorithm is the synthetic reproduction of the scene as viewed by the vision sensor with the provided intrinsic parameters, from the specified pose.

3) Synthetic aerial image generation algorithm

The synthetic aerial image generation algorithm takes every pixel location in the image. For each pixel location, the image depth λ is computed using equation (12). The next step is to determine the pixel intensity value and it depends on the value of image depth λ for the pixel under consideration.

- If $\lambda <= 0$, the pixel location in the image to be generated contains sky. The pixel intensity value is then set to sky blue {R = 79, G = 110, B = 176}.
- If $\lambda > 0$, the pixel location in the image to be generated does not include sky. The corresponding geo-location ${}^{i}p_{obj}$ in the world co-ordinates is computed using equation 11. Now the pixel intensity value {R, G, B} for the pixel location is retrieved from the GeoTIFF image database. If the corresponding pixel intensity is not available in the database its value is set to black {R = 0, G = 0, B =0}.

IV. Results and Discussion

This section presents the results obtained and discusses them in detail. The synthetic aerial image is generated using the algorithm described in section III which is implemented in MATLAB (Figure 3). The default extrinsic parameters for a downward looking camera located above the NAL are given in Table III. The default intrinsic parameters used for the camera model are indicated in Table IV.

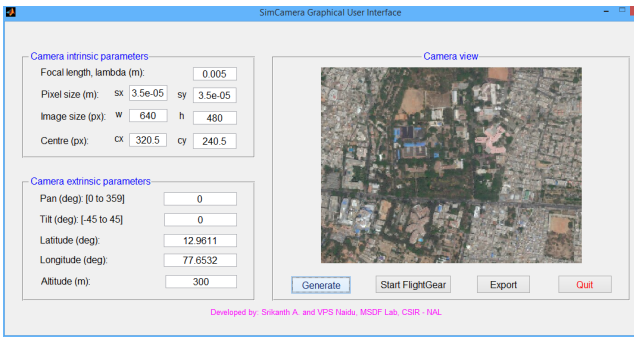


Fig. 3. Interface of the Synthetic Aerial Image Generation GUI

TABLE III. Extrinsic parameter of a downward looking camera

Extrinsic parameter	Value
Latitude, degree	12.9611
Longitude, degree	77.6532
Altitude, metre	300
Pan, degree	0
Tilt, degree	0

TABLE IV. Intrinsic parameter of the emulated camera

Intrinsic parameter	Value

Intrinsic parameter	Value
Focal length (f), metre	5e-3
Pixel scale factors (sx, sy), metre	35e-6, 35e-6
Principal point (cx, cy), pixel	320.5, 240.5
Image size (w, h), pixel	640, 480

The implementation uses the perspective camera model to determine world co-ordinates corresponding to each pixel in the camera frame, which is later converted to geographic co-ordinates (latitude and longitude). Then the corresponding pixel intensity values are read from a local GeoTIFF image database and the scene is generated. Scenes generated at different configurations (i.e., pan and tilt angles) are shown in Figure 4 (a, b, c, d, e). Clicking on the "Export" push button the parameters are exported to the file system as a '.scene' file and corresponding image as a '.png' file. The structure of '.scene' file is discussed in Appendix B.

1) Separation of Sky and Non-sky Region

The image generated by the algorithm may contain region covered by sky. We have employed a method for determining the pixels which belong to the sky region using the depth information. The method is described in section III.3 and the results obtained can be observed in Figure 4 (d, e).

2) Image Generation using FlightGear Flight Simulator

FlightGear Flight Simulator is open-source software widely used by the scientific community for simulation, modelling, analysis and visualization of aerial vehicles. We used a customized build of FlightGear with PhotoScenery integration [5]. The corresponding aircraft configuration file is modified as mentioned in Appendix C. In the SimCamera MATLAB interface the user can click on "Start FlightGear" pushbutton, which will start Flight Simulator with the selected configuration using the following command and the result is shown in Figure 4 (f, g, h, i, j). The meanings of command line options are described in "The FlightGear Manual" [3].

V. Conclusion

This paper presented a method to generate a synthetic aerial scene from a local GeoTIFF image database. The mathematics required to generate the synthetic scene are discussed in detail. The result of the implementation of the said method on MATLAB is presented.

References

[1] Guowei Cai, Jorge Dias, Lakmal Seneviratne, "A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends", Unmanned Systems Vol. 02, No. 02, 2014, pp. 175-199.

[2] D. Blake Barber, Joshua D. Redding, Timothy W. McLain, Randal W. Beard, Clark N. Taylor, "Vision-based Target Geo-location using a Fixed-wing Miniature Air Vehicle", Journal of Intelligent and Robotic Systems, Volume 47, Issue 4, December 2006, pp 361-382.

[3] Yi Ma, Stefano Soatto, Jana Kosecka, S. Shankar Sastry, "An Invitation to 3-D Vision: From Images to Geometric Models", Springer Science+Business Media, 2004.

[4] The FlightGear Manual, source: http://mapserver.flightgear.org/getstart.pdf accessed on 21-05-2015.

[5] Srikanth A, Indhu B, L Krishnamurthy, VPS Naidu, "Photoscenery for Realistic Scene Generation and Visualization in FlightGear: A Tutorial", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), Vol. 3, Special Issue 5, December 2014, pp 577-582.

[6] ArcGIS REST API – ArcGIS Services – Export Map Reference, source: http://services.arcgisonline.com/arcgis/sdk/rest/02ss/02ss00000062000000.htm accessed on 21-05-2015.

[7] QLandkarteGT, source: http://www.qlandkarte.org/ accessed on 08-12-2015

Appendix A

Consider for example, we are creating a GeoTIFF database for "NAL, Bengaluru" area. We know the latitude and longitude of this geographic location, the ortho-imagery is obtained from *ArcGIS* online service using the bounds calculated by following the steps used in [5] (Section II). The ortho-image is then downloaded using the ArcGIS REST API [6]. The following URL downloads an ortho-image of resolution 4096x4096 from ArcGIS:

http://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/export?bbox=77.625%2C+12.875%2C+77.75%2C+13.0&bboxSR=4326&size=4096%2C+4096&imageSR=4326&format=png24&transparent=false&f=image

The above procedure for downloading ortho-images is repeated for a range of latitudes and longitudes to cover the whole area where the MAV is to be flown, forming an image database. When the images are downloaded, the next step is to geo-reference the ortho-images to create a GeoTIFF database.

QLandkarteGT [7] is an open-source GIS (Geographic Information System) software which is used to geo-reference an existing ortho-image. Following are the steps employed:

- From the menu bar, select "Map" → "Edit / Create Map".
- In the middle of the right half of the screen, a line with "Source: " appears, with an empty selection box. Click on the box, and select "Convert a TIFF into GeoTiff by geo referencing it".
- Now 3 steps appear below.

1. In the Step 1,
   - Choose an input file by clicking on the icon to the right of "Input File".
   - Click on the icon next to Map projection. Use "Lon/Lat" as Projection and "WGS_1984" as Datum.
   - Set Mode as "square pixels (2 Ref. Pts.)".
2. In the Step 2,
   - Click "Add Ref" (right column), move the pin to a known reference point on the map (like a corner), and double click on "<enter coord>" in the "Coord", and enter the coordinates in the form *'12.875 77.625'*.
   - Repeat for another co-ordinate on the map.
3. In the Step 3, Click on "Start process". Now it will create the GeoTIFF file.

Appendix B

Navigation data and camera parameters can be exported from "SimCamera GUI" MATLAB interface and stored in a MAT-file with '.scene' as the file extension. When the user clicks on "Export data" in the interface, the data is exported into a '.scene' file and is internally stored as a MATLAB structure. The format of the data structure implementing the camera intrinsic and extrinsic parameters, '*cam'* is given below in Table V. The format of the data structure implementing gimbal translation and rotation, '*gimbal*' is given in Table VI.

TABLE V. DATA STRUCTURE IMPLEMENTING CAMERA INTRINSIC AND EXTRINSIC PARAMETERS IN THE '.SCENE' FILE

Field Name	Field Type	Field Description
X	double	Gimbal to Camera translation, metre
Y	double	
Z	double	
w	double	Image width, pixel
h	double	Image height, pixel
f	double	Focal length, metre
sx	double	Pixel scale factors in x and y, metre
sy	double	
Cx	double	Principal point, pixel
Cy	double	
Fx	double	Focal length in x and y, pixel
Fy	double	
hfov	double	Horizontal field of view, degree
vfov	double	Vertical field of view, degree
img	uint8 matrix (h, w, 3)	Image of the generated scene
lats	double vector (1, h x w)	Latitudes of each pixel, degree
lons	double vector (1, h x w)	Longitudes of each pixel, degree

TABLE VI. DATA STRUCTURE IMPLEMENTING GIMBAL TRANSLATION AND ROTATION IN THE '.SCENE' FILE

Field Name	Field Type	Field Description
X	double	UTM Easting of the Gimbal, metre
Y	double	UTM Northing of the Gimbal, metre
Z	double	Altitude of the Gimbal, metre
alpha	double	Gimbal pan angle, degree
beta	double	Gimbal tilt angle, degree

APPENDIX C

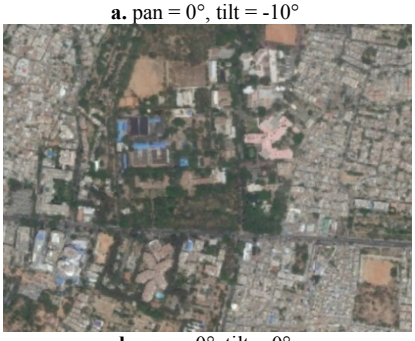
The simulated aircraft has the camera looking downwards when the aircraft is looking forward; with zero pan and tilt angles. We have added a "Custom Camera View" for this configuration by editing the concerned aircraft configuration file (in our example the file is 'ufo-set.xml').

The following lines highlighted in yellow are the modifications done in the configuration file 'ufo-set.xml' for aircraft 'ufo'.

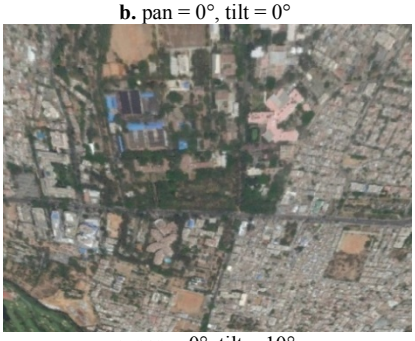
$FG_DIR\bin\data\Aircraft\ufo\ufo-set.xml

```
<PropertyList>
  <sim>
    .
    .
    .
    <view n="101">
      <name>Custom Camera View</name>
      <enabled type="bool" userarchive="y">true</enabled>
      <type>lookfrom</type>
      <internal type="bool">false</internal>
      <config>
        <from-model type="bool">true</from-model>
        <from-model-idx type="int">0</from-model-idx>
        <pitch-offset-deg>-90</pitch-offset-deg>
      </config>
    </view>
    .
    .
    .
  </sim>
</PropertyList>
```

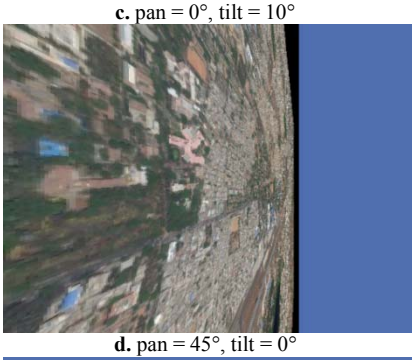
a. pan = 0°, tilt = -10°



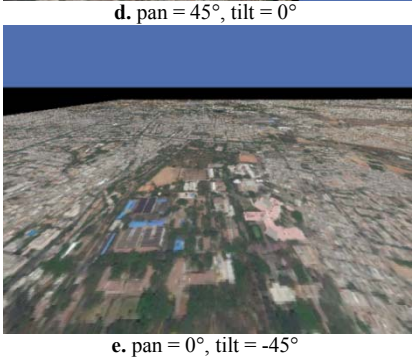
b. pan = 0°, tilt = 0°



c. pan = 0°, tilt = 10°

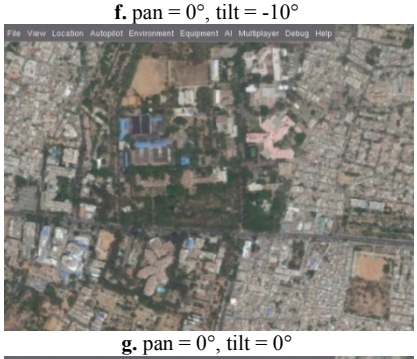


d. pan = 45°, tilt = 0°

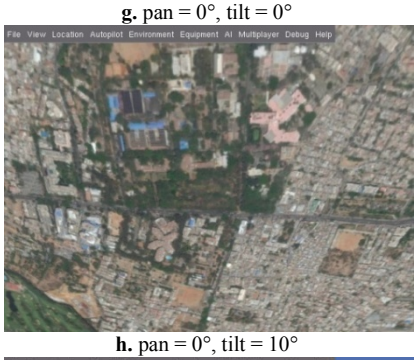


e. pan = 0°, tilt = -45°

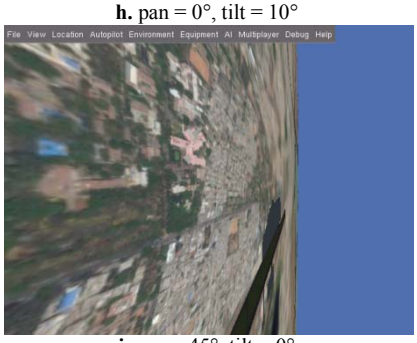
f. pan = 0°, tilt = -10°



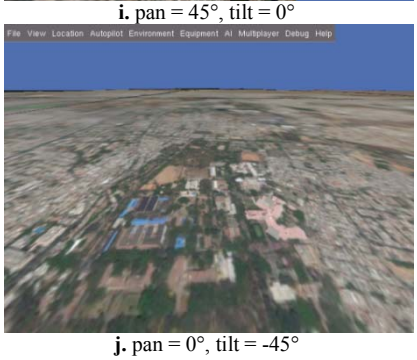
g. pan = 0°, tilt = 0°



h. pan = 0°, tilt = 10°



i. pan = 45°, tilt = 0°



j. pan = 0°, tilt = -45°

Fig. 4. The result of scene generation in MATLAB (a, b, c, d, e) and the scene rendered in FlightGear (f, g, h, i, j)